Beyond Turing: From General Computing to Domain-Specific - The "Post-Turing" Era of Domain Architectures

Zhiyu Chen *

College of Science, The University of British Columbia, Vancouver, V6L 1G1, Canada

* Corresponding Author Email: czhiyu580@gmail.com

Abstract. Alan Turing's machine model of the universal machine model laid the theory and foundation of current computers, and layers can be regarded as an essence of computer science of current generation computers, averting the ubiquity of the universal computing; however, gradually, against Moore's law and stepping into higher speed-computing needs, betraying the shortcomings of Turing's model, are come up a new structure paradigm, namely specific domain architectures. Domain architecture is a style of software development which can extracts the complex business logic and build it out as a pure 'domain model', one that is independent of technical solutions. It is a bit like preparing an exact business blueprint of a software system: the software code may truly reflect business concepts and business rules. By defining clear boundaries and responsibilities, the domain architecture makes the system highly understandable, maintainable and capable of dynamically responding to changes in frequent business requirements. This article argues that people are entering a "post-Turing era", and people gradually turn from the pursuit of universality to pursuit of efficiency and accuracy. And that success of domain-specific architectures is no mere technology nor is it a new form rather than Turing's form as represented by GPUs. Moreover, the future of computing will, sooner or later, be a heterogeneous era dominated by this new form.

Keywords: Turing Machine; Domain-Specific Architecture; Post-Turing Era; TPU.

1. Introduction

Alan Turing proposed Turing machine in 1936 and gave the concept of the theoretical machine1, which sowed seeds for modern computers with its simple structure. It defined the concept of 'computation' and also revealed the connection between computation and mathematics, philosophy, cognitive science, physics, biology, information theory, etc. With all practical applications, the Turing machine is still one of the key tools used in computational theory to study algorithms, understand their limitations and fields related to them [1]. However, the leading objective of the Turing machine's quest for "universality" has raised an undesirably large consumption of energy and time.

Today in our times of high-speed computing, the Turing machine is no longer fully applicable and a new paradigm - Domain-Specific Architecture (DSA) - is already on the rise. DSA starts from specific applications or domains' requirements, investigates characteristics of algorithms in terms of computing, memory accesses and data communications, and designs dedicated heterogeneous accelerator architectures most suitable for the application [2].

It can improve performance, reduce power consumption, and enhance performance and efficiency through customized functional cores or customized instruction set or relevant standard elements. Unlike the general-purpose computing system, it is meant to accelerate computing for specific computing domains. DSA can satisfy today's computing requirement and time requirement, and successful application of many applications further verified the effectiveness of DSA.

It heralded a new era—"the post-Turing era". By analyzing the achievements and shortcomings of Turing machine, rise of DSA, the essence and prospects of the DSA, our paper fully shows the present post-Turing era. This revolution will show clearly that today, the human race's exploration and research of the computer problem had changed from "how machines think" to "how machines come out best results concerning specific problems to think about".

2. The Achievements and Limitations of the Turing Model

Alan Turing proposed the Turing machine model. Using a simple and clear mathematical form and operation method, he successfully discovered the essence of "mechanical computation". The model consists of a read-write head, an infinitely long tape, and a state rule table. Although its construction is simple, it has a profound impact and significance. It provided the first precise mathematical definition for the vague philosophical concept of "computability". It proved that a function is "computable" when a Turing machine can start from the input and, through a finite number of mechanical operations, follow the process of the function to obtain the correct output. This is the first great achievement of the Turing model. Another achievement of the Turing machine is his Turing test. Since its proposal, the Turing test has always been like a mirror, reflecting humanity's continuous exploration of the relationship between intelligence and machines. Its greatness may not lie in providing an excellent standard for intelligent identification, but rather in completing a problem transformation. It ingeniously transforms the philosophical thorny question of "whether machines can think" into an operable and testable behavioral problem: "Can a machine imitate humans in a conversation?" This transformation allows the initial researchers in artificial intelligence to not get bogged down in the essence of "consciousness", but can focus on creating practical systems that can understand language, perform reasoning, and demonstrate knowledge. Overall, Turing delineated the boundaries of algorithmic capabilities, clarifying which problems could be solved and which ones could never be. The pinnacle of Turing's thought was the concept of the "Universal Turing Machine". This special machine takes the "description" of other Turing machines as input and can simulate their operation. This means that the hardware can be fixed, and by changing the software (the encoding on the tape), any computing task can be executed. This "store-and-retrieve simultaneously" concept is the concept of the von Neumann architecture and also became the design philosophy of all modern general-purpose CPUs. From ENIAC to today's multi-core processors, they are all engineering realizations of this universal blueprint on silicon. Over more than half a century, the improvement in computing power has been almost equivalent to the linear growth of the performance of general-purpose CPUs. People respond to all computing needs by manufacturing faster and more complex general-purpose processors. However, it is precisely this excessive pursuit of universality that has led to the inherent limitations of the Turing model as a theoretical framework. The Turing machine is only a logical model that exists in an ideal mathematical world. Its most core assumption is that an infinitely long tape cannot be realized. This assumption bypasses the core problem of resource utilization. In the real world, memory is limited, expensive, and there is a huge and insurmountable gap between the speed of accessing memory and the speed of processor logic operations, namely the "memory wall" problem. The Turing model ignores this." The 2010s were dubbed the era of "scaling up" in AI, with the goal of expanding the size of models. However, as the performance of pre-trained large models began to plateau, machine learning is entering a new stage of discovery and exploration." [3]. In addition, the Turing machine is completely deterministic and sequential. The rule table defines the unique path from one state to another, and the read-write head performs only one operation at each moment. It does not care about how long the computation takes or how much energy is consumed. It only answers "can it be computed", but cannot be calculated about the consumption of time and resources. When people pursue the ultimate performance on general CPUs, people are actually fighting against the limitations of this ideal model: people introduce cache hierarchies to simulate "fast tapes", people use out-of-order execution and branch prediction to break the shackles of sequentiality, and people build multi-core processors to attempt parallelism. These extremely complex engineering techniques are essentially in the process of patching the cracks between the ideal model and physical reality. Therefore, the Turing model points out the end of computing for us, but does not provide a path to efficient computing. Therefore, when Turing's theory no longer meets the current pursuit of efficiency, a new paradigm is needed to adjust this situation, and thus domain-specific architectures are born.

3. The Rise and Core Features of Domain-Specific Architectures

After the general computing paradigm encountered obstacles in the physical world, the development path of computing architectures began to diverge. One path was to continue making difficult marginal improvements on general-purpose CPUs, while the more revolutionary path was to abandon the "all-purpose" fantasy and turn to specific methods for specific domains. This was the rise of domain-specific architectures. The core of this can be summarized as a fundamental reversal: from requiring "software to adapt to fixed general hardware" to "customizing dedicated hardware for specific software or algorithm paradigms". This philosophical shift gave rise to the "specialized division of labor" in the computing field. The first and most successful case was the graphics processing unit (GPU). Early computers used CPUs to render graphics, and their serial nature made them inefficient. The birth of the GPU was to efficiently handle the inherent, large-scale parallel, and unified vertex and pixel transformation in graphics rendering. It adopted a many-core architecture with thousands of simplified cores, simplifying the control logic and concentrating resources on data parallel computing [4]. The success of the GPU not only revolutionized the gaming and visual industries, but more importantly, it proved to the world that by sacrificing universality, significant performance improvements in specific domains can be achieved. It is a powerful "production line" designed to handle homogeneous tasks. Google's tensor processing unit (TPU) is the first and most thorough implementation of DSA. If the GPU still has a touch of general parallel computing, then TPU is for the core operations in the machine learning field - matrix multiplication. In TPU, the hardware design directly maps software requirements: it adopts a pulsating array structure, allowing data to flowlike blood in the heart, pulsating between fixed processing units, minimizing the most energy-consuming operation of data movement [5]. TPU removes the complex branch prediction, out-of-order execution logic, and large cache system in the general CPU, and invests almost all the chip area and power consumption into pure computing units. “TPU v4 is Google's fifth domain-specific architecture (DSA) and also its third supercomputer dedicated to such ML models. The optical switch (OCS) can dynamically reconfigure its interconnection topology to enhance scalability, availability, utilization, modularity, deployment, security, power consumption and performance [6].” As a result, in AI inference and training tasks, it achieves a tens of times higher energy efficiency compared to general CPUs of the same era. Looking at GPU, TPU, and the neural network processing units that have become widespread in mobile devices, people can summarize the common features of DSA: first, they all bravely gave up universality and refused to handle tasks outside specific domains; second, they ensure the most efficient flow of data within the chip by simplifying and customizing data paths, eliminating unnecessary copying and delays; finally, and most importantly, they pursue the ultimate energy efficiency ratio, that is, the computing throughput provided per watt of power consumption. They are no longer "general-purpose computers" that can run any program, but "computing appliances" designed to solve a specific type of problem. Their success announced a new principle: in the era of exploding computing power demands, the optimal computing efficiency no longer comes from more powerful general tools, but from specially forged dedicated tools for specific tasks.

4. The Connotation of the "Post-Turing" Era: Paradigm Shift

Once domain-specific architectures have become a topic of 'technology' and buzz word of industry trend, what we see is not only a 'hardware revolution', but a 'post-Turing era', a profound revolution. The change is profoundly influencing our conception of the nature of computing, and goes much further than just modern architecture optimisation into the heart of every level. At the value system level.

Under the Turing paradigm, the ultimate value of computing lies in its universality: a machine can solve any computational problem. In the post-Turing era, the value of computing is as endomoded defining it as getting the utmost efficiency within some domain.

With this value shift, new standards and thoughts also brought with them: from high computing speeds to actual efficiency, from motivation to instruction parallelism to motivation for data flows, from motivating single-core speed to motivating join-in capabilities of the whole computing system [7].

Now, this value transformation is transforming the whole computer industry at once—from chip design to software, from architecture to business model. Behind this lie deep changes in computing ontology—computing phenomenon in the post-Turing era. Turing model is based on the ontology of computing as "symbolic computing", and all computing problems are abstracted into mechanical symbolic transformations in computing.

However, the success of the DSA tells us that the core of computing is much closer to the optimal implementation of a certain physical process. For example, a pulsating array of TPUs is actually a physical instantiation of matrix multiplication, a physical process that is equivalent to a certain mathematical calculation (a calculation to process symbolism generally), rather than one to generate a symbol tree. or what this means to us. That is, no longer do we see computing as a pure mathematical process, but as the best possible physical realization of desired information-processing task choices, which are set by physical constraints. At the system level, we are seeing a machine-to-problem transition.Traditional computing focuses on the general-purpose processor. All the problems need to be converted into a form that the processor understands. In the age after Turing, the system design starts from the specific problem to design the most suitable computing structure for the problem.

It also transforms computing systems into being heterogeneous and specialized, resulting in an algorithmic system “focused on problems”. And it has brought with it theoretical innovation as well. The old way of computing development is indeed as “hardware first” as “to have some general-purpose hardware first” and developing software that is compatible to them.

Post-Turing era also needs deep collaborative design of software and hardware, and even the evolution of a new kind of model for ‘algorithm-directed hardware’. Hardware is no longer a fixed platform, but a cooperative element which can be tweaked and shaped according to the algorithm.

Figuratively speaking, the post-Turing paradigm means that computing discipline is passing from trying to seek universal theories of computation to simply accepting diverse solutions. We no longer believe that one suitable 'correct' method of computation exists, but recognise that different problems require different possible computing paradigms.

Quantum is the most significant interpretation that the post-Turing era has brought. This scientific revolution has profound consequences on the limits and possibilities of computing. It is not just a technology revolution; it is a revolutionary reconstruction of the whole notion of computation - revolutionizing everything from how we develop more powerful general-purpose computers to how we construct the most effective solutions to specific problems. Nowadays, at this new era, the value of computing does not emanate from universality anymore but rather from its deep conditioning and final efficiency for a domain.

5. Future Prospects and Challenges of Domain-Specific Architectures

Domain-specific architectures are following their evolution through various paths, painting for all the more colorful a picture of this evolution. In technical terms, the space of technologies evolves from high-level domains towards more finely grained specialisations. Early GPUs were general-purpose parallel accelerators, but today’s TPUs and NPUs are much more deeply optimized for specific algorithms and computational patterns. In the future, DSAs will even be even sharper foci, and they will produce explicitly tailored custom architectures for autonomous driving perception, molecular drug simulations, and quantum chemistry applications [8]. “This move to more detailed specialization, therefore, is taking computer architecture into an age of precision engineering”. In contrast with general-purpose processing, which targets balanced execution, DSA design focuses on optimal execution of its target application. We envisage new architectures exploiting different algorithm features: Transformer processors that improve upon specific 'attention' operations in TFMs,

graph neural network accelerators for graph computations. New approaches are not limited to architectural customisation of computational units, but also system-wide architectural improvements on memory hierarchies and interconnects too [9]. Nevertheless, the hot-rodding success of DSAs faces big challenges. Real architectural fragmentation has serious implications, as proliferating architecturally specialized designs will exponentially increase software design, software debugging and migrations challenges. Another pillar of tension between flexibility and efficiency are obvious dilemmas: sustaining high performance of the system and keeping sufficient programmability are key problems when conceiving of DSA. And fast algorithm evolution can make good engineering of architectures largely obsolete in a short time period. Future DSA developments will focus on hardware and software co-design and system-level design optimisation. "Reconfigurable computing architectures" can attract more attention, trying to best achieve efficiency while allowing for required flexibility. Cross-platform programming models and further sophisticated compilation strategies might address the fragmentation issue [10]. After these technical considerations come with the diversification of the DSA landscape, evidence of the computing industry reaching a higher level of maturity. There, no single architecture dominates all applications; there are multiple distinct and specialised designs, each taking over a role in its domain of responsibility, generating a richly diversified computing ecosystem.

6. Conclusion

Overall, taking from this theoretical development beginning with Turing's theory to paradigm change like DSA, the solving of the problems by using computers more and more suits people's daily life.

Because of the simple and ideal model of the Turing machine, it expressed the limit of the computation, characterized the domination of general computing; whilst emergence of domain-specific architectures signifies that people are already heading into a “post-Turing” era of efficiency.

It is not only a revolution in technical architecture, but also a change in value and design philosophy, and design methods from pursuing universality to domain importance, from "machine-oriented" to "problem-oriented", separation of software and hardware to deep collaborative design

Nowadays, various special architecture schemas exhibit outstanding performances in their own particular aspects. The criterion of the value of computation is from “whether it can be computed” to “how to compute” optimally, from computing peak to actual computational efficiency. This transformation is not only a solution of the physical bottlenecks of the common computation, but also provides new opportunities for innovation of many fields. Just as the invention of Turing’s machine is in the past, even perfect theory in the future may have some problem, and there will be some new and better technologies to adapt to them.”

Looking forward, computing development will gradually evolve towards diversification and specialization. People will continually improve and refine their progress on this journey. This collision and development between theory and practice is also the source for the vitality of computational science, it will continue to lead mankind to pursue infinite exploration opportunities for intelligence and computing.

References

[1] S-Ag3. Turing machines: The universal blueprint of computation and its multidisciplinary reach. Medium. 2025. Available from: https://medium.com/@ingartsq2/turing-machines-the-universal-blueprint-of-computation-and-its-multidisciplinary-reach-71b95e2ea6d2

[2] Wang C. Domain-Specific Computer Architectures for Emerging Applications: Machine Learning and Neural Networks. 1st ed. Chapman and Hall/CRC; 2024.

[3] Li G J. Historic breakthroughs and great challenges in intelligent computing technology. Journal of Integration Technology. 2025; 14(1):1–8.

[4] Nickolls J, Dally W J, Williams J. The GPU computing era. IEEE Micro. 2010; 30(2): 56-69.

[5] Conficconi D, Del Sozzo E, Carloni F, Comodi A, Scolari A, & Santambrogio M D. An energy-efficient domain-specific architecture for regular expressions. IEEE Transactions on Emerging Topics in Computing. Advance online publication. 2022. https://doi.org/10.1109/TETC.2022.3157948

[6] Jouppi N, Kurian G, Li S, Ma P, Nagarajan R, Nai L, et al. TPU v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. Proceedings of the 50th Annual International Symposium on Computer Architecture. 2023; Article 82:1–14.

[7] Tikhonov A, Yamshchikov I P. Post Turing: Mapping the landscape of LLM evaluation. arXiv preprint arXiv:2311.02049. 2023.

[8] Krishnakumar A, Ogras U, Marculescu R, Kishinevsky M, & Mudge T. Domain-specific architectures: Research problems and promising approaches. ACM Transactions on Embedded Computing Systems, 22(2), Article 28. 2023. https://doi.org/10.1145/3563946

[9] Krishnakumar A, Ogras U, Marculescu R, Kishinevsky M, & Mudge T. Domain-specific architectures: Research problems and promising approaches. ACM Transactions on Embedded Computing Systems, 2023, 22(2), Article 28, 1–26. https://doi.org/10.1145/3563946

[10] Rothmann M, & Porrmann M. A survey of domain-specific architectures for reinforcement learning. IEEE Access, 2022, 10, 13753–13767. https://doi.org/10.1109/access.2022.3146518